

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

Annex C: Python Reference Architecture for Global Integration of the Human Equilibrium Index (HEI / IEH)

This annex provides a Python reference architecture to facilitate integration of the Human Equilibrium Index into global data systems. It is designed as an implementation-oriented technical annex, covering ingestion, normalization, indicator construction, HEI computation, uncertainty handling, API exposure, and governance-ready deployment patterns aligned with OECD-style composite-indicator practice and international statistical interoperability needs.[cite:23][cite:29][cite:32][cite:41]

1. Purpose of the annex

The objective of this annex is to provide a modular Python blueprint that can be integrated into global systems, including national statistical offices, multilateral observatories, donor platforms, and research infrastructures. The architecture is intentionally general so that it can connect to OECD, UNDP, Gallup, DHS, MICS, administrative systems, and future APIs without changing the mathematical logic of the HEI model.[cite:32][cite:35][cite:41][cite:77]

2. Recommended architecture

A production-grade HEI implementation should be divided into six interoperable layers:[cite:23][cite:29][cite:35]

1. **Connector layer** for loading data from APIs, CSV files, databases, or cloud storage.
2. **Harmonization layer** for schema mapping, validation, cleaning, and unit standardization.
3. **Indicator layer** for building primary indicators and mapping them to α_1 , α_2 , and F . [cite:32][cite:36][cite:40]
4. **Computation layer** for calculating S , B , D , and the final IEH value according to the model formula. [cite:23][cite:29]
5. **Uncertainty and validation layer** for imputation flags, Monte Carlo perturbations, and sensitivity diagnostics. [cite:23][cite:26]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

6. **Service layer** for exporting results to files, dashboards, or APIs for governments and researchers.[cite:35][cite:41]

3. Folder structure

A practical Python implementation can use the following structure:

```
hei_system/  
├── config/  
│   ├── settings.yaml  
│   └── indicator_map.yaml  
├── data/  
│   ├── raw/  
│   ├── processed/  
│   └── outputs/  
├── hei/  
│   ├── __init__.py  
│   ├── connectors.py  
│   ├── schemas.py  
│   ├── harmonize.py  
│   ├── indicators.py  
│   ├── model.py  
│   ├── uncertainty.py  
│   ├── validation.py  
│   ├── api.py  
│   └── pipeline.py  
└── run_pipeline.py
```

This modular structure follows common data-engineering practice and supports transparency, testability, and long-term governance.[cite:23][cite:29][cite:35]

4. Python reference code

4.1 Data classes and configuration

```
from dataclasses import dataclass  
from typing import Dict, List, Optional
```

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

```
@dataclass
class HEIPParameters:
    tau: float = 0.08
    k: float = 50.0
    m: float = 10.0
    f_exponent: float = 1.5
```

```
@dataclass
class IndicatorWeights:
    alpha1: Dict[str, float]
    alpha2: Dict[str, float]
    f: Dict[str, float]
```

These classes store the key calibration parameters and weight definitions in a transparent and auditable way, which is consistent with OECD/JRC guidance on composite indicators.^{[cite:23][cite:29]}

4.2 Connector layer

```
import pandas as pd
from pathlib import Path

class DataConnector:
    def load_csv(self, path: str) -> pd.DataFrame:
        return pd.read_csv(path)

    def load_parquet(self, path: str) -> pd.DataFrame:
        return pd.read_parquet(path)

    def load_api_json(self, url: str) -> pd.DataFrame:
        return pd.read_json(url)

    def save_output(self, df: pd.DataFrame, path: str) -> None:
        Path(path).parent.mkdir(parents=True, exist_ok=True)
        if path.endswith('.csv'):
            df.to_csv(path, index=False)
        elif path.endswith('.parquet'):
            df.to_parquet(path, index=False)
        else:
```

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

```
raise ValueError('Unsupported output format')
```

This layer allows the system to connect to heterogeneous sources, including OECD and UN-style open data endpoints, flat files, and data lakes.[cite:32][cite:41][cite:77]

4.3 Harmonization layer

```
import numpy as np
```

```
class Harmonizer:
```

```
def rename_columns(self, df: pd.DataFrame, mapping: Dict[str, str]) -> pd.DataFrame:
    return df.rename(columns=mapping)
```

```
def normalize_01(self, series: pd.Series, min_val=None, max_val=None) -> pd.Series:
    min_val = series.min() if min_val is None else min_val
    max_val = series.max() if max_val is None else max_val
    if max_val == min_val:
        return pd.Series(np.ones(len(series)), index=series.index)
    return (series - min_val) / (max_val - min_val)
```

```
def fill_missing(self, df: pd.DataFrame, method: str = 'median') -> pd.DataFrame:
    if method == 'median':
        return df.fillna(df.median(numeric_only=True))
    if method == 'ffill':
        return df.ffill()
    raise ValueError('Unsupported missing-data method')
```

This layer standardizes variables onto the 0–1 scale required by the model and supports explicit handling of missing data, which is essential for international comparability.[cite:23][cite:26][cite:41]

4.4 Indicator construction layer

```
class IndicatorBuilder:
```

```
def weighted_average(self, df: pd.DataFrame, weights: Dict[str, float]) -> pd.Series:
    cols = list(weights.keys())
    w = np.array([weights[c] for c in cols], dtype=float)
    values = df[cols].astype(float).to_numpy()
    return pd.Series((values * w).sum(axis=1) / w.sum(), index=df.index)
```

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

```
def build_alpha1(self, df: pd.DataFrame, weights: Dict[str, float]) -> pd.Series:  
    return self.weighted_average(df, weights)
```

```
def build_alpha2(self, df: pd.DataFrame, weights: Dict[str, float]) -> pd.Series:  
    return self.weighted_average(df, weights)
```

```
def build_f(self, df: pd.DataFrame, weights: Dict[str, float]) -> pd.Series:  
    return self.weighted_average(df, weights)
```

This step translates observed indicators into the conceptual structure of family capacity, social embeddedness, and structural support.[cite:32][cite:36][cite:40]

4.5 HEI computation layer

```
import numpy as np
```

```
class HEIModel:
```

```
    def __init__(self, params: HEIParameters):  
        self.params = params
```

```
    def compute_s(self, alpha1: pd.Series, alpha2: pd.Series) -> pd.Series:  
        return (alpha1 + alpha2) / 2.0
```

```
    def compute_b(self, alpha1: pd.Series, alpha2: pd.Series) -> pd.Series:  
        diff = (alpha1 - alpha2).abs()  
        tau, k = self.params.tau, self.params.k  
        return pd.Series(  
            np.where(diff <= tau, 1.0, np.exp(-k * (diff - tau) ** 2)),  
            index=alpha1.index,  
        )
```

```
    def compute_d(self, ds: pd.Series) -> pd.Series:  
        m = self.params.m  
        return 1.0 / (1.0 + np.exp(-m * (ds - 0.5)))
```

```
    def compute_hei(self, alpha1: pd.Series, alpha2: pd.Series, f: pd.Series, ds: pd.Series) -> pd.DataFrame:  
        s = self.compute_s(alpha1, alpha2)  
        b = self.compute_b(alpha1, alpha2)
```

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

```
d = self.compute_d(ds)
hei = b * (f ** self.params.f_exponent) * s * d
return pd.DataFrame({
    'alpha1': alpha1,
    'alpha2': alpha2,
    'F': f,
    'S': s,
    'B': b,
    'D': d,
    'IEH': hei,
})
```

This code directly encodes the mathematical logic of the model and can be reused in national, regional, or global pipelines.[cite:23][cite:29]

4.6 Time-series and adaptability support

```
def compute_relative_change(series: pd.Series, periods: int = 1) -> pd.Series:
    return series.pct_change(periods=periods).replace([np.inf, -np.inf], np.nan).fillna(0.0)
```

This function supports rolling computation of dS , which is required to estimate adaptability under the HEI framework.[cite:23][cite:35]

4.7 Uncertainty layer

```
class UncertaintyEngine:
    def monte_carlo_perturbation(self, df: pd.DataFrame, cols: List[str], pct: float = 0.05, n: int = 100):
        simulations = []
        for _ in range(n):
            sim = df.copy()
            for c in cols:
                noise = np.random.uniform(1 - pct, 1 + pct, size=len(sim))
                sim[c] = sim[c] * noise
            simulations.append(sim)
        return simulations
```

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

Monte Carlo perturbation is appropriate for testing sensitivity to measurement error and missing-data uncertainty, which is explicitly recommended for robust composite-index design.[cite:23][cite:26]

4.8 Validation layer

```
from scipy.stats import spearmanr
```

```
class ValidationEngine:
```

```
    def rank_correlation(self, hei_df: pd.DataFrame, external_df: pd.DataFrame, hei_col='IEH', ext_col='HDI'):  
        merged = hei_df[[hei_col]].join(external_df[[ext_col]], how='inner')  
        coef, pval = spearmanr(merged[hei_col], merged[ext_col], nan_policy='omit')  
        return {'spearman': coef, 'p_value': pval, 'n': len(merged)}
```

This layer supports convergent validation against external indices such as HDI and related development metrics.[cite:25][cite:44][cite:77]

4.9 Pipeline orchestration

```
class HEIPipeline:
```

```
    def __init__(self, connector: DataConnector, harmonizer: Harmonizer, builder: IndicatorBuilder, model: HEIModel):  
        self.connector = connector  
        self.harmonizer = harmonizer  
        self.builder = builder  
        self.model = model
```

```
    def run(self, df: pd.DataFrame, weights: IndicatorWeights) -> pd.DataFrame:  
        df = self.harmonizer.fill_missing(df, method='median')  
        alpha1 = self.builder.build_alpha1(df, weights.alpha1)  
        alpha2 = self.builder.build_alpha2(df, weights.alpha2)  
        f = self.builder.build_f(df, weights.f)  
        s_temp = (alpha1 + alpha2) / 2.0  
        ds = compute_relative_change(s_temp, periods=1)  
        return self.model.compute_hei(alpha1, alpha2, f, ds)
```

This orchestration pattern makes the system easy to schedule in Airflow, Prefect, Dagster, or cloud-native batch services.[cite:35][cite:41]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

4.10 Minimal execution example

```
if __name__ == '__main__':
    df = pd.DataFrame({
        'income_security': [0.70, 0.55, 0.40],
        'childcare_access': [0.65, 0.50, 0.35],
        'caregiver_mental_health': [0.60, 0.48, 0.42],
        'community_participation': [0.62, 0.45, 0.30],
        'social_cohesion': [0.66, 0.50, 0.28],
        'basic_services': [0.90, 0.72, 0.50],
        'family_policy_support': [0.75, 0.55, 0.30],
        'childcare_system_strength': [0.72, 0.52, 0.33],
        'social_protection_strength': [0.80, 0.58, 0.40],
    }, index=['CountryA', 'CountryB', 'CountryC'])

    weights = IndicatorWeights(
        alpha1={
            'income_security': 0.4,
            'childcare_access': 0.3,
            'caregiver_mental_health': 0.3,
        },
        alpha2={
            'community_participation': 0.3,
            'social_cohesion': 0.4,
            'basic_services': 0.3,
        },
        f={
            'family_policy_support': 0.4,
            'childcare_system_strength': 0.3,
            'social_protection_strength': 0.3,
        },
    )

    params = HEIPParameters()
    pipeline = HEIPipeline(DataConnector(), Harmonizer(), IndicatorBuilder(), HEIModel(params))
    result = pipeline.run(df, weights)
    print(result.round(4))
```


This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

This example demonstrates how the architecture can be embedded into existing statistical or policy platforms with only minor adaptation to local schemas.[cite:32][cite:35]

5. API exposure pattern

A practical global deployment should expose IEH outputs through a lightweight API so that dashboards, governments, and research teams can query results programmatically.[cite:35][cite:41]

```
from fastapi import FastAPI
import pandas as pd

app = FastAPI(title='HEI API')
RESULT_PATH = 'data/outputs/hei_latest.csv'

@app.get('/health')
def health():
    return {'status': 'ok'}

@app.get('/hei/latest')
def hei_latest():
    df = pd.read_csv(RESULT_PATH)
    return df.to_dict(orient='records')

@app.get('/hei/country/{country_code}')
def hei_country(country_code: str):
    df = pd.read_csv(RESULT_PATH)
    out = df[df['country_code'] == country_code.upper()]
    return out.to_dict(orient='records')
```

This pattern is suitable for integration with dashboards, observatories, and donor systems, and can be extended with authentication, versioning, and uncertainty metadata.[cite:35][cite:41][cite:77]

6. Deployment recommendations

For global use, the Python implementation should be deployed with the following principles:[cite:23][cite:29][cite:35]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

- **Containerization:** use Docker for reproducible execution across cloud and institutional environments.
- **Workflow orchestration:** use Airflow, Prefect, or Dagster for scheduled data refresh and traceable ETL runs.
- **Data versioning:** store raw and processed snapshots with immutable timestamps.
- **Model versioning:** record parameter sets, weight maps, and code versions for each published HEI release.
- **Auditability:** attach quality flags, imputations, and uncertainty intervals to every published observation.
- **Interoperability:** use country codes, standardized schemas, and documented metadata compatible with statistical agencies and multilateral repositories.[cite:32][cite:41][cite:77]

7. Final technical conclusion

The HEI model can be operationalized in Python using a modular architecture that is simple enough for research teams and extensible enough for global institutions. Its mathematical core is straightforward to encode, while its real implementation challenge lies in data harmonization, calibration, uncertainty management, and governance—exactly the areas where modular Python pipelines are strong and where OECD- and UN-oriented statistical systems already operate.[cite:23][cite:29][cite:32][cite:41]

LICENSE AND CONDITIONS OF USE.

The content is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

By using this content and model, you agree to:

Provide appropriate credit, include a link to the above license, and indicate if changes were made.

Not use the material or any derivative works for commercial purposes.

Distribute any adaptations or derivative works only under the same CC BY-NC-SA 4.0 license.

Not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Data logging and trade secrets policy.

The design, implementation, and deployment of this model, as well as any derivatives, do not authorize the collection, storage, or exploitation of IP addresses, unique device identifiers, or other technical traceability data for commercial profiling, targeted advertising, or any other economic exploitation.

Any system implementing this model must limit the logging and retention of IP addresses and other identifiable data strictly to what is necessary to:

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full "License and Conditions of Use" notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

(a) maintain technical security and service integrity; and

(b) comply with applicable legal obligations (for example, any minimum log retention required by law).

It is prohibited to use this model or its derivatives to:

capture, store, or exploit trade secrets, proprietary know-how, or confidential information of users or third parties for any purpose other than providing the technical service described in this documentation; or train or improve closed, commercial systems using data that contains confidential information of third parties without their prior, explicit, written consent.

---+EOD+---